Early Experiences with the Myricom-X2000 Switch on an SMP Beowulf-Class Cluster for Unstructured Adaptive Meshing

CHARLES D. NORTON AND THOMAS A. CWIK

Jet Propulsion Laboratory
California **Institute** of **Technology**
MS 168-522, 4800 Oak Grove Drive
Pasadena, CA 91109-8099 USA

Contact Author: Charles.D.Norton@jpl.nasa.gov

Abstract

We explore the current capabilities of the recently released Myricom X2000 witch, using MPICH over the GM communication layer, for message passing on a 2-way SMP Pentium III Beowulf-Class cluster: Performance measurements on the network indicate that data transfer rates of approximately 190 Mbytes/s for ping-pong tests, and 240 Mbytes/s for fill-duplex exchange tests, can be achieved for messages as large as 32 Mbytes. The performance varies depending on how processors communicate; either within an SMP node or across nodes. Results with the network for parallel unstructured adaptive refinement of 3D tetrahedral meshes show noticeable performance improvement when compared to 100BaseT Ethernet. Furthermore, when compared to traditional systems such as the SGI Origin 2000, the combination of this fast network with high performance SMP processors demonstrate that Beowulf-Clusters compare favorably with such systems—even when communication intensive applications are used.

Keywords:
Performance Evaluation, Beowulf Cluster, AMR, Network, Myrinet, SMP

1. Introduction

Beowulf-Class clusters have demonstrated that scalable parallel computing can be achieved, at low cost, through the use of commodity off-the-shelf (COTS) parts. The COTS approach allows one to configure a system using the latest hardware and software that is available, and affordable, at the time. As time goes by, one has the option to reconfigure a system as new products are announced and released. Many cluster components may be freely available, such as system software, while others, including advanced microprocessors, are constantly under competitive pricing pressures to remain affordable.

The network interconnect, however, allows great flexibility regarding the communication performance one expects verses the amount one is willing to pay. Although 100BaseT Ethernet cards are not expensive the ~11 MBytes/s upper bound can hinder the performance of clusters containing many fast SMP processors with large amounts of main memory. (It is interesting to observe how improvements in system software and microprocessor performance bring added pressure to consider fast networking to maintain a balanced system.)

Within the JPL High Performance Computing Group we have a series of clusters, one of which is quite powerful. This cluster contains **26** compute nodes, and one front end node, of dual-processor 800 MHz Pentium III's—a total of 54 processors in all. Each node has 2 GBytes of RAM available giving a system with 104 GBytes of main memory with **41.6** GFlops of computation. We recently replaced our 3COM SuperStackII switch and 100BaseT Ethernet network with Myricom's new **32-port X2000** networking hardware. More details about Myricom's technology including architecture specifications, **software,** algorithms, and products are available at their web site and elsewhere [1, 6].

We will explore our experiences and evaluate the performance impact **this** hardware introduces for our system. The results will be compared to experiences before the new network was introduced, and to a more traditional system (the **SGI** Origin **2000**) for adaptive meshing simulations that typically stress CPU, memory, and communication performance.

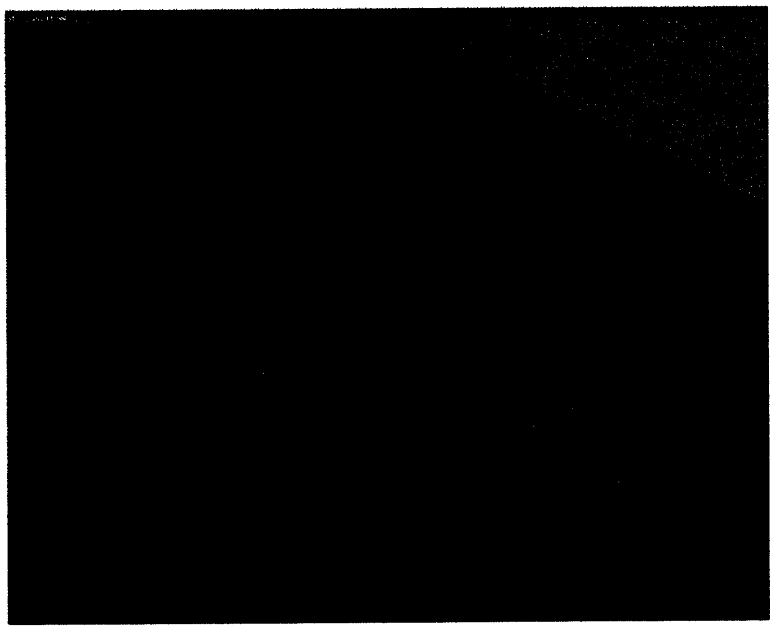


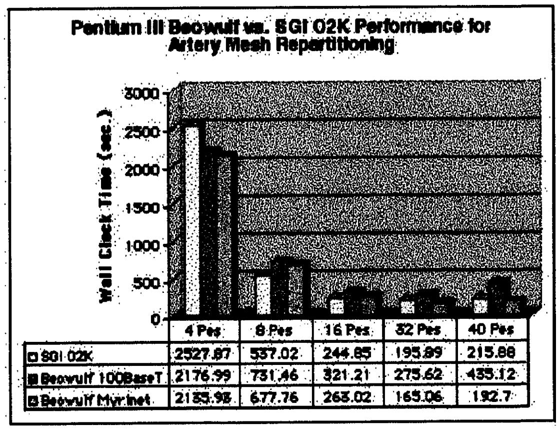
Figure 1. Repartitioning and migration of artery mesh segment using PYRAMID AMR Library.

2. Characterizing Communication-Intensive Applications

Many physics-based numerical applications are fundamentally irregular, meaning that the solution process is largely determined at run-time and the communication requirements are non-uniform, although generally predictable. Parallel adaptive methods fall into **this** category **and** software for these techniques require very high performance systems for large problems.

We have developed software to handle parallel unstructured adaptive mesh refinement for finite element applications [5, 7]. **This** tool allows large triangular and tetrahedral meshes to be loaded, adaptively refined with automatic mesh quality control, load balanced, and migrated among the processors using high level object-based library commands. Since parallel adaptive mesh refinement potentially involves working with many millions of elements great effort is used to minimize communication and to ensure that transmitted messages are **as** large as possible. On a cluster, managing communication becomes even more important since applications may use networks that are significantly slower than those found on most traditional supercomputing systems.

Figure 1 shows a segment of a large tetrahedral artery blood flow mesh segment. The original geometry was provided by Taylor et. al [8] and the initial mesh was generated by the Scientific Computation Research Center at Rensselaer Polytechnic Institute [2]. The mesh contains 1.1 million elements where **our PYRAMID** adaptive mesh refinement library was used for repartitioning, load balancing, and mesh migration. Processor partitioning is illustrated by color.



	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes
□ SGI O2K	2527.87	537.02	244.85	195.89	215.88
▨ Beowulf 100BaseT	2176.99	731.46	321.21	275.62	435.12
▨ Beowulf Myrinet	2135.98	677.76	263.02	165.06	192.7

Figure 2. Performance for mesh repartitioning and migration of the artery mesh.

Performance comparisons for this problem using 100BaseT Ethernet and Myrinet on the cluster, as well **as** the NUMA architecture of the SGI O2K, indicate the benefit of Myrinet, shown in figure **2.** This is largely a communication-based benchmark where a variety of message sizes are used. As we will see, however, the performance tradeoffs vary based on the problem solved. Nevertheless, this initial benchmark indicates that the Beowulf cluster *can* compete on par with traditional systems even for communication intensive applications.

3. Configuring MPICH-GM under Linux

Although cluster technology has generally stabilized, when new components are integrated into an existing system problems can develop. At this date, our Beowulf cluster is based on Redhat Linux 6.2 with Kernel version 2.4.4 using MPICH-GM version 1.2..7 with patches. Initially, however, our hardware vendor installed kernel version 2.2.x which did not support the features of our system. Also, Myricom's MPICH-GM version 1.2..5, based on MPICH 1.2.0, had problems that were not immediately known. This introduced delays, but fortunately Myricom support has been very active in working with us to address and correct problems.

Regarding the initial kernel installation we found that the vendor did not realize that certain configuration options required by our system were not available. In fact, the highmem region of memory was not addressable so we experienced unnecessary memory swapping that also interfered with the memory requirements of the GM communication layer. Fortunately, most of the defaults for kernel version 2.2.4 satisfied our system requirements so only a few extra details required attention. This included specifying that we had SMP processors, that the amount of real memory was between 1-4MB, and that the highmem region of memory should be addressable.

There are a number of ways to configure MPICH-GM for communication among MPI processes. If shared memory support is not specified then messages are sent over the Myrinet. If the `-shared-memory-support` and `--disable-direct-copy` options are used then SMP communication with a 2-copy protocol is used. Alternatively, if only `-shared-memory-support` is used, where direct-copy is enabled by default, then a 1-copy SMP protocol is used. There were a number of problems related to these configuration options and they are still being addressed. Nevertheless, using various patches from Myricom, a good combination of stability and performance is achieved with shared memory support enabled and direct copy disabled causing a 2-copy protocol to be invoked. Other configurations generate errors reported by the `_gm_sent()` routine, or in user programs, on occasion. For architectures that support memory registration the `-gm-can-register-memory` option will allow a registration/zero-copy mode. This option did not currently give a performance improvement for our simulations and we experienced less stability.

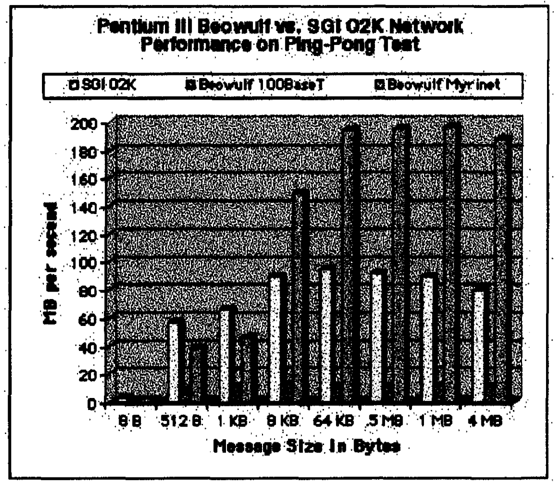
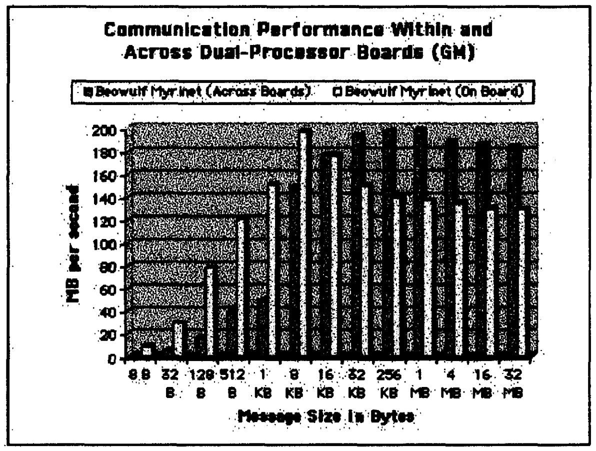


Figure 3. MBytes/s transmitted in ping-pong tests between two processors.

Figure 3 shows the results of a network ping-pong test for the Myrinet installation compared to previous results for 100BaseT and the SGI Origin 2000. Incidentally, the MPI implementation on the Origin uses the global shared memory to implement message passing. When a processor requires data the packets are sent over the CrayLink so the latency to access memory becomes a critical part of a performance metric on this machine in addition to good cache management.

The improvement for our cluster is significant where neighbor processors that are not on the same board are used. This certainly had an impact on the artery mesh migration problem in figure 2. While we are very close to peak speed for 100BaseT for MPICH implemented over GM we are only at about 54% of the *peak* speed of the GM communication layer alone (reported rated at 2.8Gbits/s). The average latency is still quite low, about 18μ sec. for processors that share a CPU board and 37μ sec. for processors across CPU boards. You may notice a crossover point between the SGI 02K and Myrinet communication performance. The result in figure 3 uses processors across CPU boards.

Others have reported that there can be an overhead associated with MPICH over GM, and that on some systems the MPI implementation can take away as much as 23% of the peak bandwidth of Myrinet [3]. Nevertheless, other benchmarks have indicated that 149 MBytes/s with 10μ latency have been achieved with older versions of the Myrinet hardware [6], so our new equipment does show a modest improvement over these results for MPICH over GM.



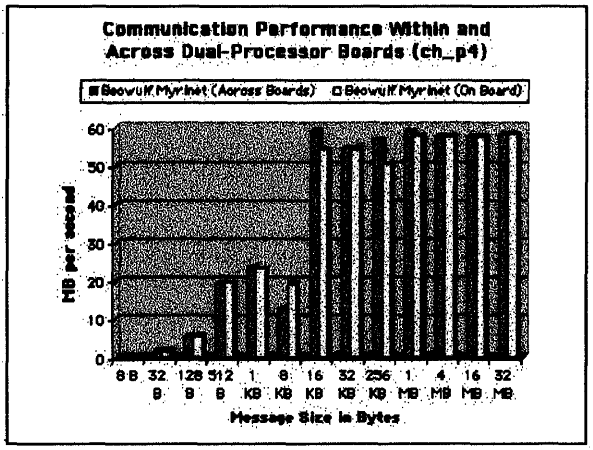


Figure 4. MBytes/s transmitted in ping-pong tests between two processors on the same board, and across boards, when using both the ch_gm and ch_p4 devices under MPICH.

4. Evaluating Functionality and Performance

Our primary interest is to examine the effect of a network upgrade for a communication intensive application. However, before examining how the X2000 hardware performs on our adaptive meshing simulations we should take a closer look at network performance in an SMP environment. Additionally, since the GM layer is not completely stable at this time, one may also want to allow Myrinet to emulate an IP network where the ch_p4 device will run over TCP/IP as a replacement. Understanding the performance implication of this decision is also important for applications.

The ch_gm device results in figure 4 show that good performance is possible, particularly for large messages. There is a cross-over point where communication between processors on the same CPU board falls behind processor communication across boards for messages larger than 16 KBytes. We have no speculation as to why this cross over occurs at this point. We do know that for our adaptive meshing problem we regularly send messages as large as 35 MBytes in size so our simulations will select processors one at a time, one board at a time.

Regarding the ch_p4 device, commonly associated with the IP protocol, there are steps one *can* take to actually send IP traffic over Myrinet and that is what we have done. This is generally a matter of building MPI with the ch_p4 device and ensuring that the Myrinet configuration files and system host tables are set up properly. This gives an improvement over the Ethernet cables and the performance is essentially independent of the processor configuration chosen. This arrangement allows one to use the cluster reliably, but not up to its full potential.

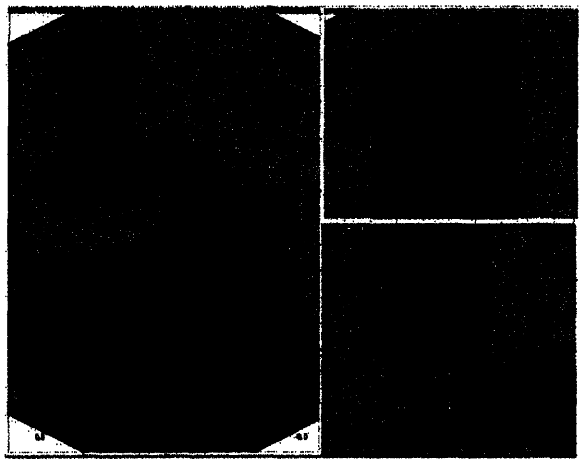
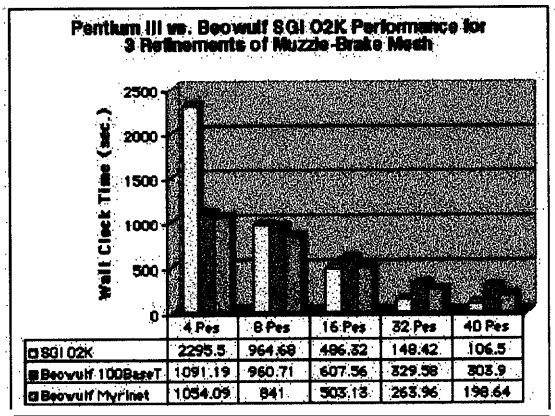


Figure 5. Muzzle-brake shock tube mesh with initial mesh partitioning and redistribution among eight processors.

4.1 Analyzing the Muzzle-Brake

Figure *5* shows a muzzle-brake shock tube mesh with its initial partitioning and redistribution among processors. We repeated the simulations from [7] using the new Myrinet X2000 and have included the combined results in figure *6*. The initial mesh contains just 34,214 elements, but after three adaptive refinements the new mesh, shown in figure 7, contains 1,264,443 elements.

What is clear is that for a small number of processors the



	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes
▢ SGI 02K	2295.5	964.68	486.32	148.42	106.5
▦ Beowulf 100BaseT	1091.19	960.71	607.56	329.58	303.9
▨ Beowulf Myrinet	1054.09	841	503.13	263.96	198.64

Figure 6. Performance after three adaptive refinements of the muzzle-brake mesh.

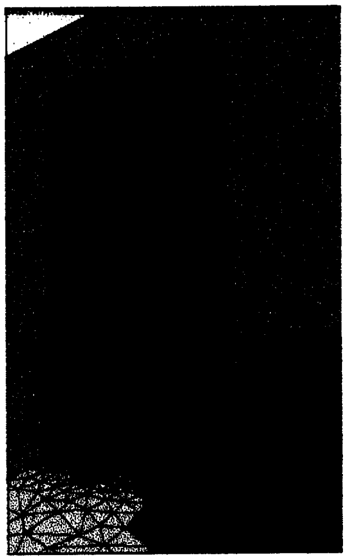


Figure 7. Performance after three adaptive refinements of the shaft section of the muzzle-brake mesh.

Beowulf cluster performs much better than the Origin for this problem. As the number of processors is increased all configurations show some scalability, but the Origin outperforms the cluster. One would suspect that a fast Myrinet network would show an even greater improvement over the 100BaseT Ethernet. This is misleading, however, because for this specific mesh the communication performance is not dominant.

A breakdown of the time spent in mesh migration, which includes partitioning and load balancing, compared to creating new elements by adaptive refinement shows that much more time is spent in the AMR process. On the Beowulf cluster using Myrinet ~ 182s and ~46s are spent in creating new elements and migrating them respectively. Similarly, –98s and ~22s are respectively spent in these stages on the Origin. In fact, once the coarse elements have been redistributed the new elements are created locally so no communication is required. This implies that improvements in the network will not significantly impact overall performance for this specific problem.

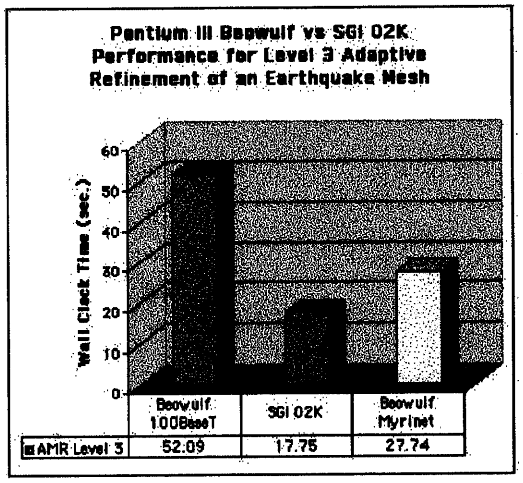
4.2 Analyzing an Earthquake Mesh

Figure 8 shows the performance for an earthquake mesh generation where communication is more dominant. These results show a nearly 50% performance improvement for the cluster under Myrinet. The initial mesh only contains 1,316 elements where 554,141 elements are created after three refinements. This example shows how a change in the problem description can impact performance for applications that have irregular characteristics.

Another important point, however, regards improvements in algorithm design for communication on clusters. This is also shown in figure 8 where the migration time is measured for 8 processors based on old and new communication algorithm techniques. The original algorithms set up communication schedules based on processors sending and receiving messages directly as needed. That approach assumed that good performance can be achieved by matching communication operations exactly. Since the communication schedule is irregular processors managed non-uniform message passing activities, but the volume of data transferred was limited to only what was required among processors that communicate.

The new communication algorithm is much more scalable for large systems in that a tree-based pair-wise exchange algorithm is used. This algorithm works on any number of processors, not just a power of two. It is unique in that is guarantees that a minimal number of exchanges will be performed–meaning that the algorithm can determine if a processor has already received the data it needs and skip communication operations as necessary. The volume of data tends to grow with such algorithms as exchanges are performed, but this algorithm can also determine when data need not be included in future pairwise exchanges and it will remove such data from future operations. It is designed to take advantage of networks supporting full-duplex communication.

Although for tightly-coupled networks this algorithm will work well, it is very well structured for clusters that have slower networks, as seen in figure 8 for the 100BaseT Beowulf network. When Myrinet is applied the results



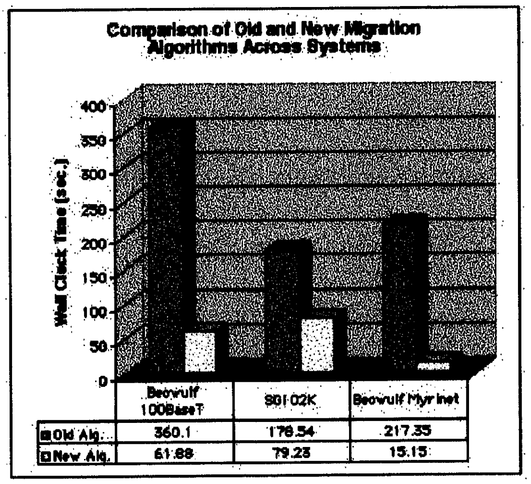


Figure *8.* Performance after three adaptive refinements of an earthquake mesh among **32** processors where Myrinet network upgrade affects overall performance. Also shown is the migration stand-alone timing due to message passing algorithm improvements for **8** processors.

are even more dramatic. *All* of the performance results in this paper are based on using the new communication algorithms.

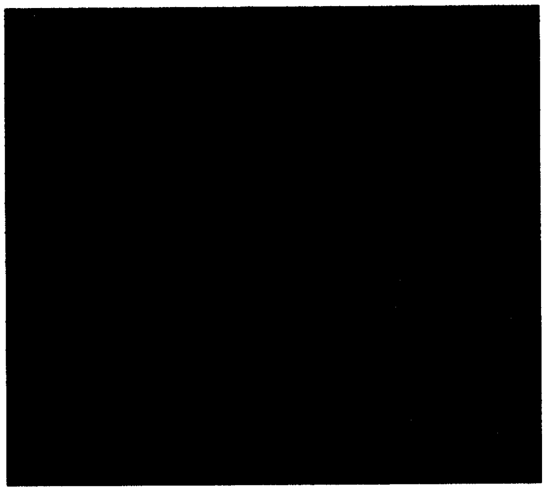


Figure 9. Adaptive refinement, repartitioning, and migration of the artery mesh segment containing **1.8** million elements.

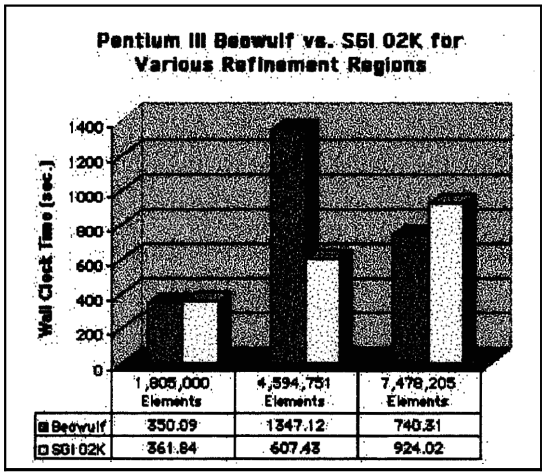
4.3 Analyzing an Artery Mesh

Returning to our artery mesh figure 9 shows the adaptive refinement of a small section that creates **1.8** million elements from the initial mesh of **1.1** million elements. Figure **10** shows again how performance can vary between the Myrinet-based Beowulf cluster and the SGI Origin 2000. In this example we used **50** processors since the GM messaging layer would occasionally fail in message send operations when all **52** processors were used.

The first instance in figure **10** adaptively refined a **small** region and analysis showed that the time spent in migration and adaptive refinement were nearly **equal across** both **machines**. In the second **instance** the region refined was increased, creating about **4.5** million elements. **For** the cluster both the migration and adaptive refinement time increased significantly, but more time was spent in adaptive refinement than migration. This pattern was also true for the SGI Origin, but both migration and refinement were faster than on the cluster.

In the third instance the entire artery was refined creating **7.4** million elements. In this case, where one would expect the cluster to perform poorly, it actually outperformed the SGI Origin. Analysis showed that the migration time for the cluster was slower than for the **Origin** while the adaptive meshing time was faster for the cluster than for the Origin. Overall, **this** gave the cluster better performance in this case.

We also examined the message passing structure more closely for these problem instances. Although the new pairwise message exchange algorithm was used for mesh mi-



	1,805,000 Elements	4,594,751 Elements	7,478,205 Elements
▨ Beowulf	350.09	1347.12	740.31
□ SGI 02K	361.84	607.43	924.02

Figure 10. Performance comparison for adaptive refinement of artery mesh for various region sizes on **50** processors.

gration we decided to analyze the mapping for processor communication exactly. **This** included measuring, for each processor, the number of other processors that need to send data **as** well **as** the number of elements each processor must receive in order to achieve load balance during migration. Neither of these measurements allowed specific conclusions to be drawn about the performance differences among these problems cases. In fact, the data interaction was fairly well balanced for all problem sizes under both architectures. There were instances, however, where the number of elements a small number of processors must receive for load balance was about ***5*** times larger than the average, but **this** was characteristic of all test cases.

This would suggest that characteristics of the problem may determine performance much more than features of the network, and that for irregular problems this is difficult to characterize.

Although it is risky to speculate for irregular problems, we are aware that the balance between CPU **speed** and the time to fetch **data** from memory does play a role in performance. The pipelined super-scalar architecture of the R12000 processors on the Origin **are** only clocked at **300** Mhz, but the processor to memory communication interaction is very well balanced leading to good single-node performance. The Pentium architecture, historically, **has** not been balanced as well potentially causing memory accesses **to** fall behind high CPU clock rates. This has been seen in calculations involving highly structured matrix operations where the ability to control how **data** is accessed can be controlled. Although adaptive meshing is very irregular by nature, the possibility exists that for certain problems the CPU to memory interaction dominates performance more **than** the network on a cluster.

5. Conclusion

As one would expect, the upgrade of a network mainly benefits problems with large message passing requirements. ***On*** point-to-point ping-pong **tests** using Myricom's MPICH under the **GM** messaging layer we can achieve about 190 MBytes/s, or about 1.5 GBits/s, for sufficiently large messages. On an SMP where shared-memory communication is possible, we **also** observed a crossover where communication **on** the dual-CPU board falls behind communication **across** boards for large messages. Ultimately the ability to sustain network performance for large messages is more important for users running large applications.

Another useful performance measurement **is** the performance of the network where multiple **pairs** of processors are exchanging messages simultaneously. This indicates network performance under a load. Figure 11 shows rather dramatic results of this test for 16 processors using Myrinet and MPICH-GM. The error bars **also** show **the** minimum and maximum range of data transfers values we experienced during **this test.** For large messages, the average transmission **rates** clearly fall toward the low side.

Our new migration algorithms perform such pair-wise exchanges and we ***can*** see that under a load the performance drops significantly. We have also included results from the ping-pong test when multiple processors are performing the test simultaneously. This shows that for one-way (half-duplex) message transfers occurring simultaneously **that** performance drops to about 100 MBytes/s for average size messages and falling sharply for large messages. More importantly, for full-duplex message transfers between processors where we would expect good performance, we may **only see** about **30** MBytes/s **on** average. **This** most likely is a contributing factor to the performance of our adaptive meshing application and may also explain why improvements over our Ethernet connection are limited. In these tests processor **pairs** communicate across boards, not within the **same board.**

When the pair-wise exchange test is performed again, where dual-processors communicate within the same board, the results improve. For the ping-pong tests an average of 125 MBytes/s is achieved. The fullduplex exchange test shows an average of 120 MBytes/s.

One caveat regarding performance comparisons between the Beowulf SMP Cluster and the SGI Origin 2000 for our adaptive mesh problem is that the simulation results **are** not precisely identical. In particular, the ParMetis partitioner applied in the dynamic load balancing stage, produces different partitionings on each machine [4]. (This is a side-

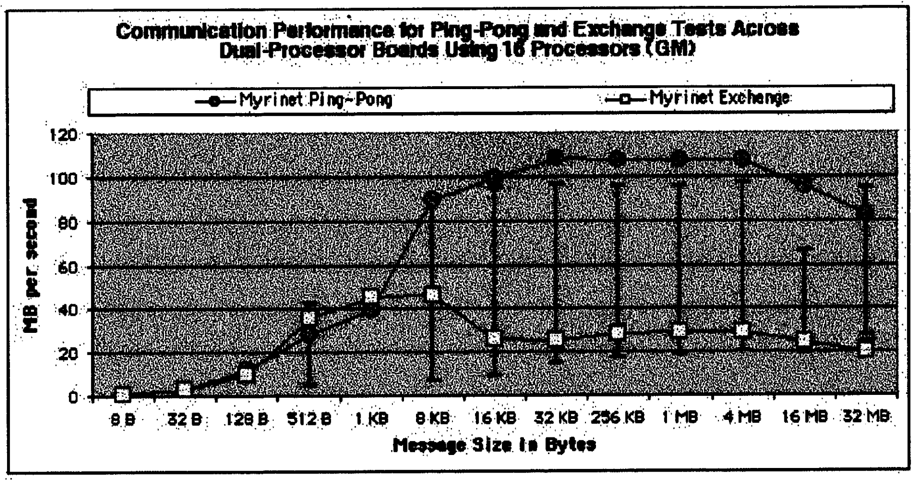


Figure 11. Performance comparison for network swap test where processor pairs exchange data simultaneously.

effect of the random number generation process.) This will affect the adaptive refinement process and may affect comparative timings, but this should be not too significant.

We have also noticed that performance varies among these machines as simulations are repeated. As the SGI Origin is a shared resource this effect is expected. The Beowulf runs were performed in isolation, but timings also varied under Myrinet and MPICH-GM. In some cases, on both systems, the variance was significant. We always used the best performance numbers in our reported results.

Actually getting to the point where programs could run fairly reliably using the X2000 switch, MPICH-GM, and RedHat Linux was not a small effort. As early users we can run applications on our system, but some instability still exists. We have seen improvements resulting from access to software fixes and patches from Myricom support. As this continues we suspect that we should see increased reliability.

In the end, the network is just one contributor to the combination of factors that affect performance and usability of the cluster. Our experience is that for functionality tests good performance can be achieved using X2000 but the effective performance improvement depends largely on characteristics of the application.

6. Acknowledgment

We appreciate several helpful discussions with Edith Huang and Gary Gutt of the JPL Supercomputing Group, members of of the High Performance Computing Group, as well as the Myricom Technical Support team. This work was performed at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

References

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet – A Gigabit-per-Second Local-Area Network. *IEEE Micro*, 15(1):29–36, February 1995.

[2] Joseph E. Flaherty and James D. Teresco. Software for Parallel Adaptive Computation. In Michel Deville and Robert Owens, editors, *P m . 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Lausanne, 2000. IMACS. Paper 174-6.

[3] J. Hsieh, T. Leng, V. Mashayekhi, and R. Rooholamini. Architectural and Performance Evaluation of GigaNet and Myrinet Interconnects on Clusters of Small-scale SMP Servers. In *Proc. SC'2000*, Dallas, Texas, November 04–10 2000. IEEE Computer Society. CD-ROM.

[4] G. Karypis, K. Schloegel, and V. Kumar. ParMetis: Parallel Graph Partitioning and Sparse Matrix Ordering Library. Technical report, Dept. of Computer Science, U. Minnesota, 1997.

[5] J. Z. Lou, C. D. Norton, and T. Cwik. A Robust Parallel Adaptive Mesh Refinement Software Package for Unstructured Meshes. In *Proc. Fifth Zntl. Symp. on Solving Irregularly Structured Problems in Parallel*, 1998.

[6] Myricom, Inc. *Myricom Creators of Myrinet*, 2001. http://www.myri.com.

[7] C. D. Norton, J. Z. Lou, and T. A. Cwik. Status and Directions for the PYRAMID Parallel Unstructured AMR Library. In *15th International Parallel and Distributed Processing Symposium*, San Francisco, CA, April 23-27 2001. Irregular 2001 Workshop. CD-ROM.

[8] Charles A. Taylor, Thomas J. R. Hugues, and Christopher K. Zairns. Finite Element Modeling of Blood Flow in Arteries. To appear, *Comp. Meth. in Appl. Mech. and Engng.*, 1999.